AI_HW4

109550050

May 6 2022

Part I. Experiment Results (the score here is included in your implementation):

1. taxi.png

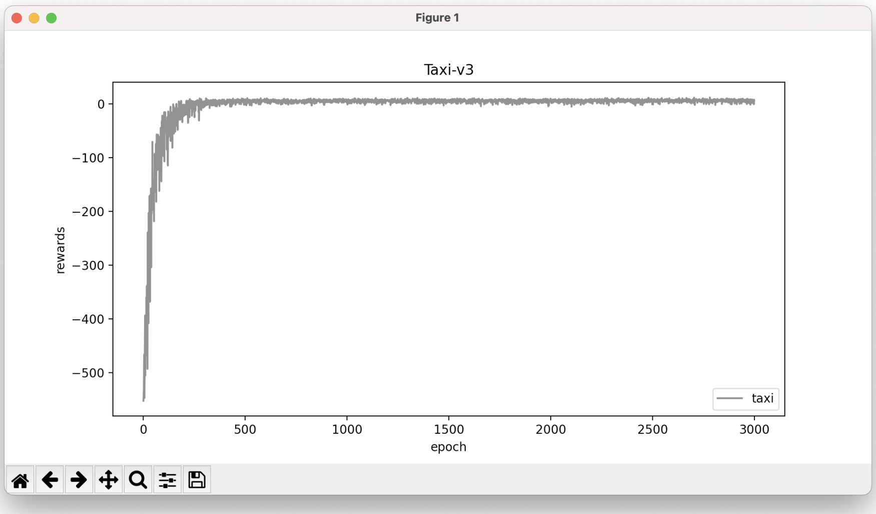


Figure 1: Taxi

2. cartpole.png

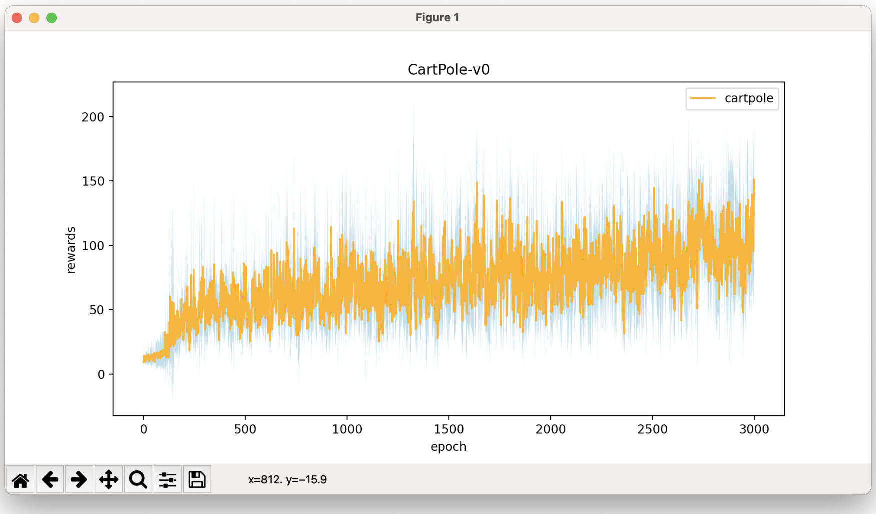


Figure 2: Cartpole

3. DQN.png

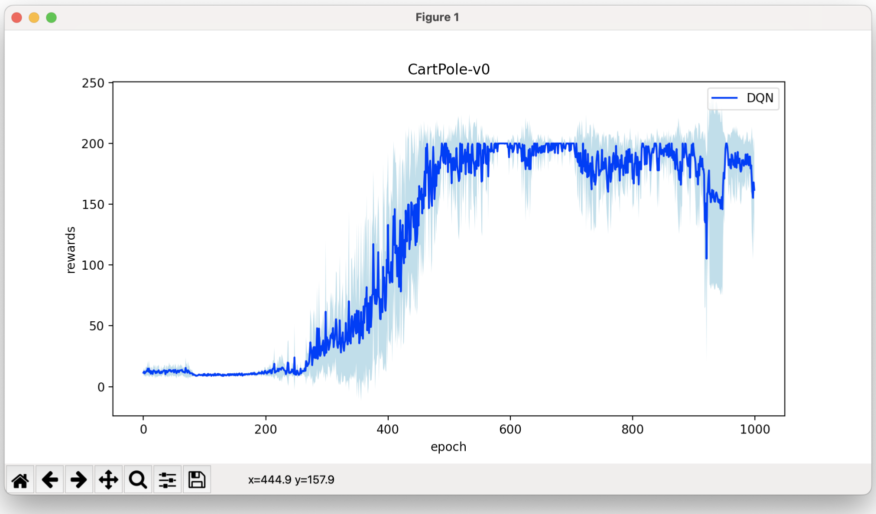


Figure 3: DQN

4. compare.png

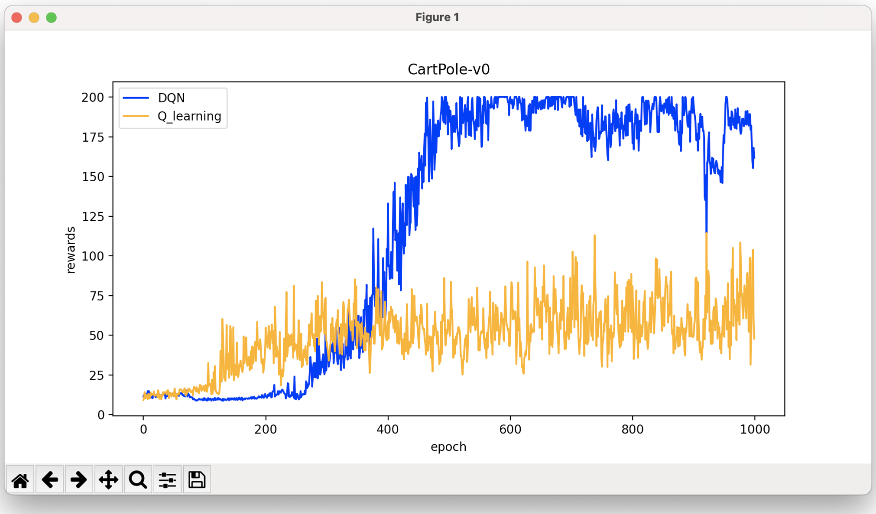
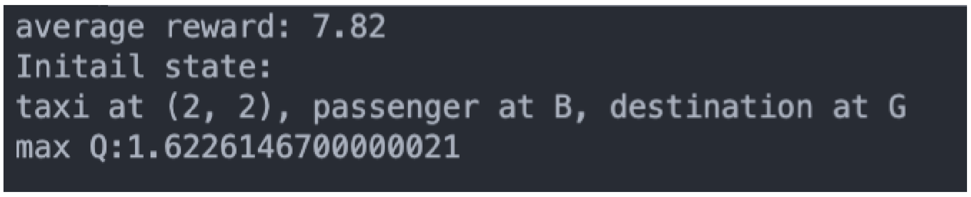


Figure 4: Compare

Part II. Question Answering (50%):

1. Calculate the optimal Q-value of a given state in Taxi-v3 (the state is assigned in google sheet), and compare with the Q-value you learned (Please screenshot the result of the "check_max_Q" function to show the Q-value you learned). (4%)

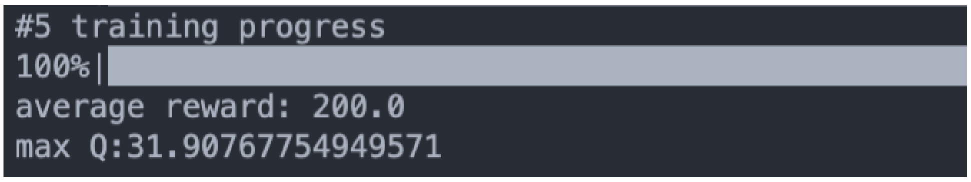
$$\begin{aligned} Q_{opt}(s,a) &= Reward(s,a,s') + \gamma * V_{opt}(s') \\ &= (-1) + 0.9[(-1) + (-1)*0.9 + (-1)*0.9^2 + ... + (-1)*0.9^7 + 20*0.9^8] \\ &\approx 1.6224 \end{aligned} \tag{1}$$



```
average reward: 7.82
Initail state:
taxi at (2, 2), passenger at B, destination at G
max Q:1.6226146700000021
```

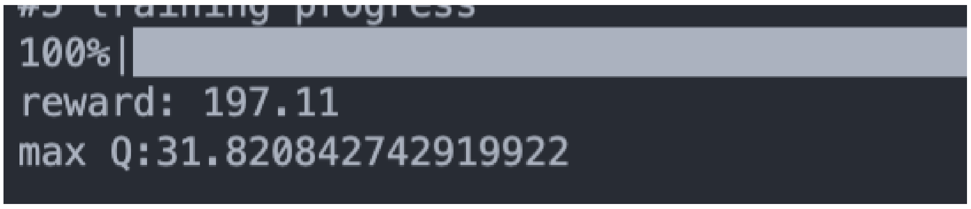
2. Calculate the max Q-value of the initial state in CartPole-v0, and compare with the Q-value you learned. (Please screenshot the result of the "check_max_Q" function to show the Q-value you learned) (4%)

$$\begin{aligned} Q_{opt}(s,a) &= Reward(s,a,s') + \gamma * V_{opt}(s') \\ &= 1 + 0.97[1 + 1*0.97 + 1*0.97^2 + ... + 1*0.97^{198}] \\ &\approx 33.258 \end{aligned} \tag{2}$$



```
#5 training progress
100%|
average reward: 200.0
max Q:31.90767754949571
```

Figure 5: part2



```
#5 training progress
100%|
reward: 197.11
max Q:31.820842742919922
```

Figure 6: part3

3.

a. Why do we need to discretize the observation in Part 2? (2%)

If we don't discretize the observations, there will be infinite kinds of observations. In other words, there will be infinitely different states. This will make the size of Qtable be infinite, which is impossible for us to implement in the real world. Therefore, we need to use discrtizing to reduce the size of Qtable.

b. How do you expect the performance will be if we increase "num_bins"? (2%)

This means that we will generate more available states and we can transfer observations into states more precisely. In this situation, I expect the performance will be better for more detailed information a state carries. Suppose that there are only states can record the integer cart positions originally , such as 1, 2... But if we have more states that can record .5 cart positions, such as 1.5, then it can more precisely describe some observations like position = 1.4. Therefore, this way can help the Qtable reflect the suitable Q values.

c. Is there any concern if we increase "num_bins"? (2%)

This causes the size of Qtable larger. In other words, we need more memory to store Qtable and more time to train the Qtable.

4. Which model (DQN, discretized Q learning) performs better in Cartpole-v0, and what are the reasons? (3%)

DQN is better because it throws the observations into the neural network to compute its q value. That is, DQN can maintain infinite states. Once we have a observation, neural network can give us an exclusive q value for this observation. In contrast, discretized Q learning has only finite states, which means the observations in an interval will be categorized into the same discrete value. Hence, DQN can do more precise decisions than discretized Q learning can do.

5.

a. What is the purpose of using the epsilon greedy algorithm while choosing an action? (2%)

This ensures we will explore the actions that we may never try. Sometimes, we find a descent Qtable that can generate great results. However, there may be other better Qtable that we don't find it. We use this algorithm to explore the actions that we seldom choose to find other possible Qtables.

b. What will happen, if we don't use the epsilon greedy algorithm in the CartPole-v0 environment? (3%)

As 5a mentioned, if we find a descent reaction to the environment, we will always choose this as our best policy and do not try other possible policies. This way, we may lose the opportunity to find a better policy than what we have now.

c. Is it possible to achieve the same performance without the epsilon greedy algorithm in the CartPole-v0 environment? Why or Why not? (3%)

If we use other algorithms to explore the environment, it's possible. We need a way to help us explore the environment randomly to find some possibilities. This can make sure we find a great policy by exploring throughout the environment. However, if we don't use any exploration method, it's almost impossible. At first, I think maybe we can get a not bad policy. Contrary to my expectation, when I do some trials to do the cartpole without epsilon, the results are terrible. It only gets an average 9.4 with max q value is 6.92. This proves that it's almost impossible to find a great policy without epsilon. But, I think there is an exception, if we are lucky enough, we may find the best policy at first time. In that case, maybe we can do it.

d. Why don't we need the epsilon greedy algorithm during the testing section? (2%)

In the testing section, we want to test if our Qtable is great. Therefore, we need to do the policy according Qtable. The epsilon is only uesd to help us to find other better policies and modified the Qtable. In the testing section, we don't need to modify Qtable and just want to test it. Therefore, we don't need epsilon algorithm.

6. Why is there "with torch.no_grad():" in the "choose_action" function in DQN? (3%)

We use this to avoid unnecessary backpropagations and calculations. When we do the choose actions function, we should not do backpropagations and calculate the gradients. This can avoid this and also reduce the computations so that we can reduce the resource consumption. Out of curiosity, I try to commend this line in the codes and implement it again. However, it seems that no significant changes. Therefore, in my opinion, this can only reduce some calculation.(But in my trial, there are only few seconds difference)

7.

a. Is it necessary to have two networks when implementing DQN? (1%)

No, it's not necessary.

b. What are the advantages of having two networks? (3%)

With a target network, the process of learning will be more stable. Without target network, we can only use evaluate network as the standard. However, this standard is always changing, we may always chase the standard, which makes the learning process unstable. Besides, we update the target network after learning for 100 times can also guarantee we have a more stable learning because we don't affect by the extreme values.

c. What are the disadvantages? (2%)

I can't find some significant disadvantages. I think it may be we need more resources when having two networks, such as memories and computations.

8.

a. What is a replay buffer(memory)? Is it necessary to implement a replay buffer? What are the advantages of implementing a replay buffer? (5%)

Replay buffer is a memory that used to store the experience the agent interacts with the environment. It's not necessary. It's more like a skill we use to achieve some benefits we expect. Using replay buffer, we can make use of our experiences. Getting experiences from interacting with the environment costs a lot. Therefore, we can make efficient use of these previous experiences by learning it multiple times(because it's expensive). The other benefit is this way can ensure we choose a batch of diverse experiences. While learning, we don't expect to choose a set of the experiences with the similar properties, which causes bad learning process.

b. Why do we need batch size? (3%)

Batch size indicates how many experiences we choose from the replay buffer. This way, we can reuse our experiences by selecting many experiences one time.

c. Is there any effect if we adjust the size of the replay buffer(memory) or batch size? Please list some advantages and disadvantages. (2%)

In an acceptable range, if we enlarge the size of the replay buffer, we can make use of the experiences and make the experience diverse. This way, the training will be more stable and we can get a general result. If we make the size smaller, our training will be less stable and make specified situations perform better than others. However, these are based on reasonable adjusts. If we make the size too small, we can't make use of the experience. If we make it too large, it will cause more computations.

9.

a. What is the condition that you save your neural network? (1%)

$$Score_{rec} = 0.9 * score + 0.1 * Score_{rec} \tag{3}$$

where $Score_{rec}$ is a indicator to decide whether we should save neural network and *score* is the scores we get from one episode. I use a global variable $Score_{max}$ to record the maximum value I have got until now. Then we save once $Score_{rec}$ is greater than $Score_{max}$

b. What are the reasons? (2%) At first, I only use the reward as my condition. However, I find this way we may save some unstable networks. To deal with it, I try to add the score before this episode into account with the way similar to the way we use to update Q value. I want to use the learning rate as my coefficients originally, but I find this combinations of coefficients can make the results better with my seed. Therefore, I choose these coefficients.

10. What have you learned in the homework? (2%)

Originally, I really have no ideas about reinforcement learning because it's really too abstract to imagine. After implementing this homework, I think I have a clear blueprint about what it is and what it does. However, this is also the most difficult homework compared to the three homework before. The most difficult part is to realize what Q learning and DQN do and figure out the pytorch library.